

ML 09: ANOMALY DETECTION, RECOMMENDER SYSTEM

이번시간엔 *anomaly detection* 과 *recommender system* 을 배운다.

Anomaly Detection

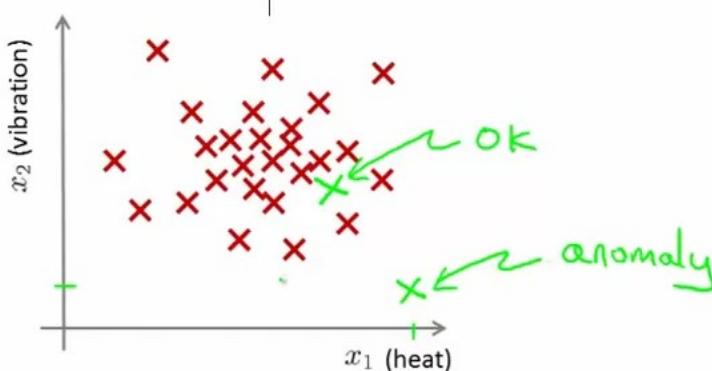
Anomaly detection example

Aircraft engine features:

- x_1 = heat generated
- x_2 = vibration intensity
- ...

Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

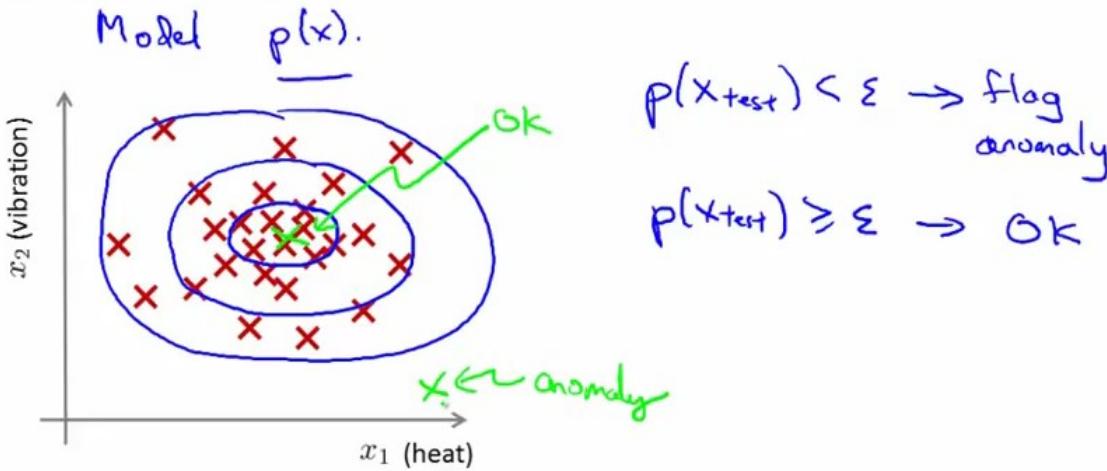
New engine: $\underline{\underline{x_{test}}}$



Density estimation

- Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

→ Is x_{test} anomalous?



(<http://blog.csdn.net/linuxcumt1>)

*anomaly*는 정상집단에서 떨어진 데이터라 보면 된다. 공장에서 품질이 떨어지는 제품을 골라낼 때 사용할 수 있는데, 위 그림은 비행기 엔진 공장을 예로 들어 설명한다.

데이터로부터 $p(x)$ 를 만들어, 검사할 데이터가 *threshold*를 넘는지 안 넘는지 검사해 *anomaly*로 판정할 수 있다.

참고로, *anomaly*가 너무 많으면, *false positive*가 높은 것인데 이 때는 *threshold*를 줄이면 된다.

Anomaly detection example

→ Fraud detection:

→ $x^{(i)}$ = features of user i 's activities

→ Model $p(x)$ from data.

→ Identify unusual users by checking which have $\underline{p(x) < \varepsilon}$

x_1
 x_2
 x_3
 x_4
 $p(x)$

→ Manufacturing

→ Monitoring computers in a data center.

→ $x^{(i)}$ = features of machine i

x_1 = memory use, x_2 = number of disk accesses/sec,

x_3 = CPU load, x_4 = CPU load/network traffic.

...

$p(x) < \varepsilon$

(<http://blog.csdn.net/linuxcumt1>)

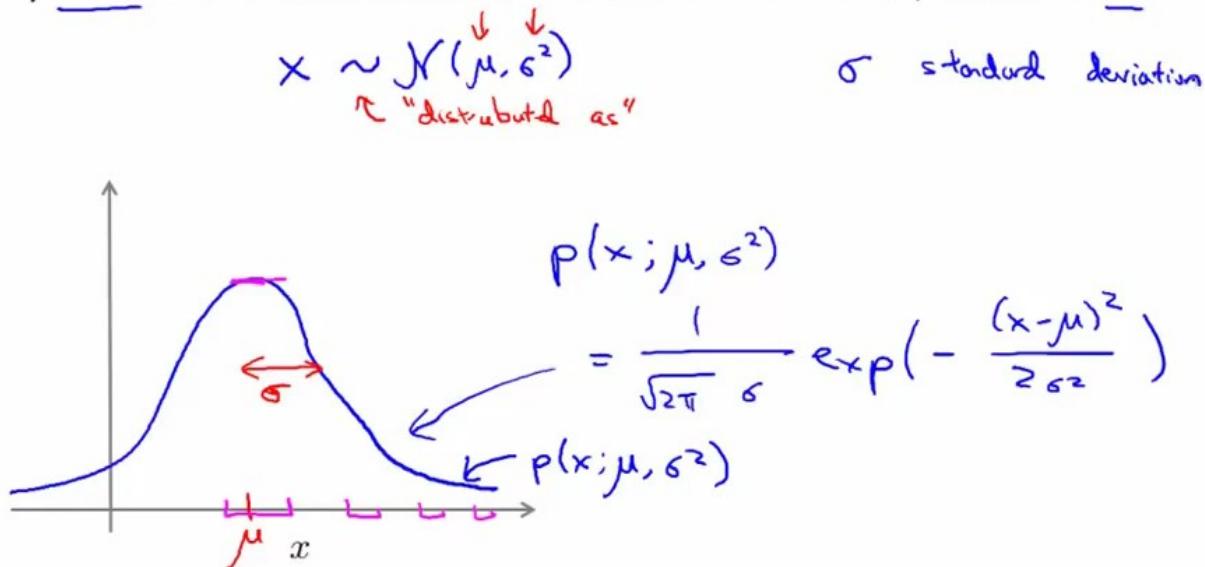
anomaly detection 은 *fraud detection* 에 많이 사용된다. 데이터로부터 모델 $p(x)$ 를 만들고 *unusual user* 를 검사하기 위해 $p(x) < e$ 인지 검사하면 된다.

이외에도 항공기 엔진 예제처럼 제품의 품질 관리나, 데이터 센터에서의 노드 과부하 탐지등에 사용할 수 있다.

Gaussian Distribution

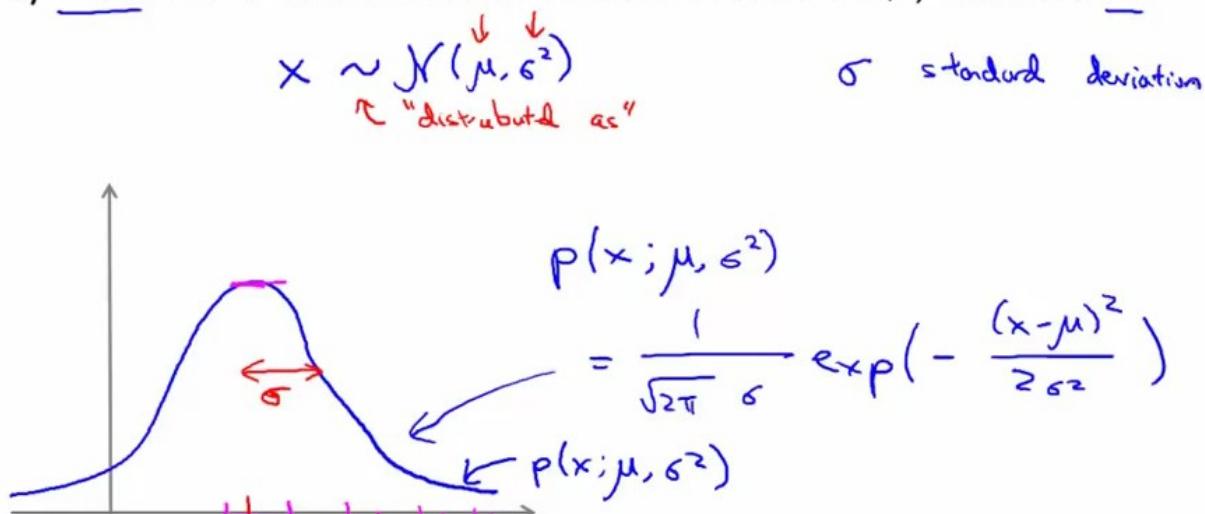
Gaussian (Normal) distribution

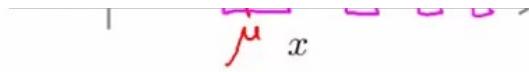
Say $x \in \mathbb{R}$. If x is a distributed Gaussian with mean μ , variance σ^2 .



Gaussian (Normal) distribution

Say $x \in \mathbb{R}$. If x is a distributed Gaussian with mean μ , variance σ^2 .





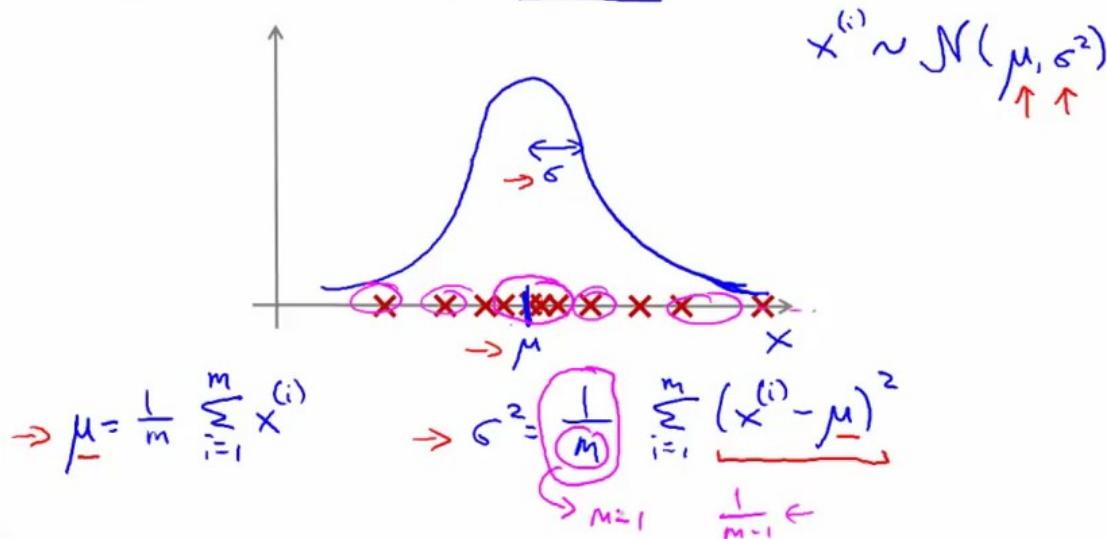
(<http://blog.csdn.net/linuxcumt1>)

gaussian density 공식은

$$P(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

Parameter estimation

→ Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}$



(<http://blog.csdn.net/linuxcumt1>)

평균과 분산은

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)^2$$

... $i=1$

Anomaly Detection Algorithm

Density estimation

→ Training set: $\{x^{(1)}, \dots, x^{(m)}\}$

Each example is $x \in \mathbb{R}^n$

$\rightarrow p(x)$

$$= p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) p(x_3; \mu_3, \sigma_3^2) \cdots p(x_n; \mu_n, \sigma_n^2) \leftarrow$$

$$= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

$$\sum_{i=1}^n i = 1+2+3+\cdots+n$$

$$\prod_{i=1}^n i = 1 \times 2 \times 3 \times \cdots \times n$$

(<http://blog.csdn.net/linuxcumt1>)

각 feature 가 가우시안 분포를 따른다고 하면,

$$p(x)$$

$$= p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) \cdots p(x_n; \mu_n, \sigma_n^2)$$

$$= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)$$

이렇게 가정하려면, 각 feature 가 독립적이어야 하지만 실제로는 독립적이지 않더라도 어느정도 동작한다. 이 때

$$\mu_i = \frac{1}{m} \sum_{i=1}^m x_i^{(i)}$$

$$\bar{x}_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$

Anomaly detection algorithm

- 1. Choose features x_i that you think might be indicative of anomalous examples. $\{x_1^{(1)}, \dots, x_n^{(m)}\}$
- 2. Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$
 - $\rightarrow \mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$
 - $\rightarrow \sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$
- 3. Given new example x , compute $p(x)$:

$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^n \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

Anomaly if $p(x) < \varepsilon$

(<http://blog.csdn.net/linuxcumt1>)

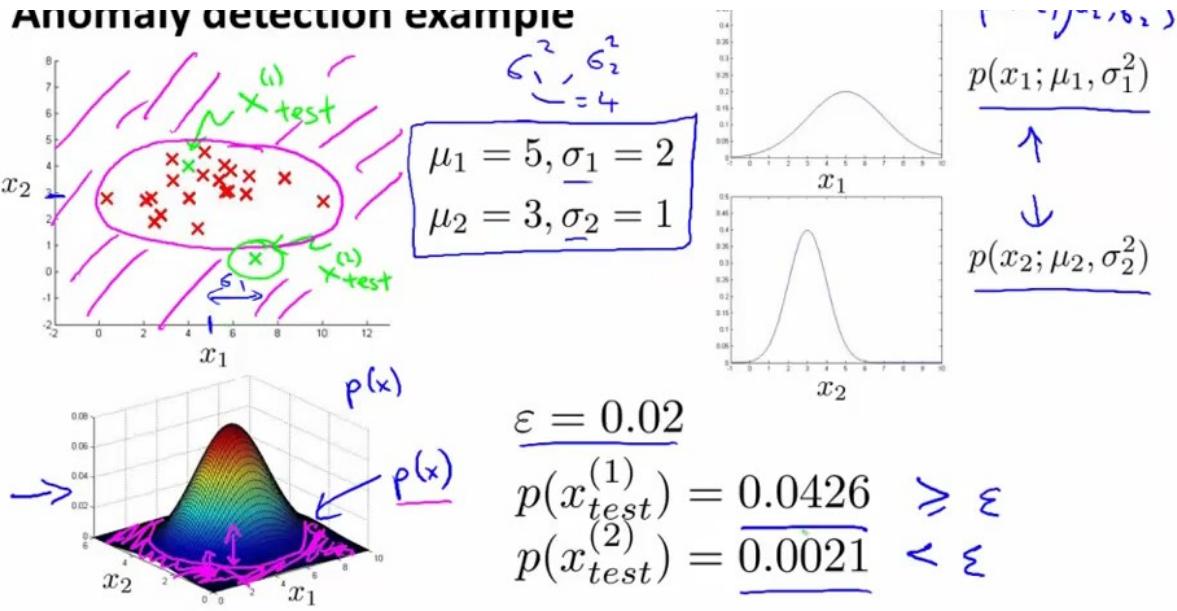
따라서 $p(x)$ 는 아래 식이 된다. $p(x)$ 는 feature 가 나올 확률로 이해하면 된다. 이 때 $p(x)$ 가 상당히 작으면, 평균에 가깝지 않은 feature 가 많이 나왔다는 뜻이므로 *anomaly* 라 볼 수 있다.

$$p(x)$$

$$= \prod_{j=1}^n \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

$$\rightarrow p(x) = p(x_1; \mu_1, \sigma_1^2) \cdot p(x_2; \mu_2, \sigma_2^2) \cdots p(x_n; \mu_n, \sigma_n^2)$$

Anomaly detection example



(<http://blog.csdn.net/linuxcumt1>)

두 feature x_1, x_2 의 가우시안 분포를 3차원으로 조합하면 $p(x)$ 가 좌측 하단 3차원 원뿔의 높이가 된다.

Evaluating Anomaly Detection

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc.), making decisions is much easier if we have a way of evaluating our learning algorithm.

- Assume we have some labeled data, of anomalous and non-anomalous examples. ($y = 0$ if normal, $y = 1$ if anomalous).
- Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ (assume normal examples/not anomalous)
- Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$
- Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

$y=1$

Aircraft engines motivating example

- [10000] good (normal) engines
- [20] flawed engines (anomalous) $\frac{2}{50}$ $y=1$
- Training set: [6000] good engines ($y=0$) $p(x) = p(x_1; \mu_1, \sigma^2_1) \dots p(x_n; \mu_n, \sigma^2_n)$
 CV: [2000] good engines ($y=0$), [10] anomalous ($y=1$)
 Test: [2000] good engines ($y=0$), [10] anomalous ($y=1$)

Alternative:

- Training set: [6000] good engines
- CV: [4000] good engines ($y=0$), [10] anomalous ($y=1$)
- Test: [4000] good engines ($y=0$), [10] anomalous ($y=1$)

(<http://blog.csdn.net/linuxcumt1>)

anomaly 를 잘 나타낼거 같은 *feature* 를 골라내고, 이를 이용해 모델을 만든다.

우리가 가진 데이터가 *anomaly* 를 알려주는 y 가 있다면, 위 그림처럼 *training set* 으로 *non-anomalous* 을 이용하고, *CV*, *Test Set* 으로 나머지를 반반씩 분할하면 된다.

즉 *good example* 로 모델을 만들고, *anomaly* 가 섞여있는 *cv*, *test set* 으로 평가한다.

Algorithm evaluation

- Fit model $p(x)$ on training set $\{x^{(1)}, \dots, x^{(m)}\}$ $(x_{\text{test}}^{(i)}, y_{\text{test}}^{(i)})$
- On a cross validation/test example x , predict

$$y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \\ 0 & \text{if } p(x) \geq \varepsilon \end{cases} \quad y=0$$

Possible evaluation metrics:

- - True positive, false positive, false negative, true negative
- - Precision/Recall CV
- - F_1 -score \leftarrow

Can also use cross validation set to choose parameter ε \leftarrow

(<http://blog.csdn.net/linuxcumt1>)

이 때 *skewed classes* 이기 때문에 ($y = 0$ 이 대다수, $y = 1$ 은 희박) 단순히 정확도로 평가하긴 좀 무리가 있다. *precision*, *recall*, *f1 score* 등을 이용해 평가해야 한다.

threshold 인 e (엡실론) 를 고르기 위해 *cross validation* 을 이용할 수 있다. *f1 score* 를 최대화 하는 e 를 고른다거나.

Anomaly Detection vs Supervised Learning

y 값이 있는 데이터라면, 왜 *supervised learning* 을 이용하지 않을까?

Anomaly detection	vs.	Supervised learning
<ul style="list-style-type: none"> → Very small number of positive examples ($y = 1$). (0-20 is common). → Large number of negative ($y = 0$) examples. $P(x)$ → Many different “types” of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; → future anomalies may look nothing like any of the anomalous examples we’ve seen so far. 		<p>Large number of positive and negative examples. ←</p> <p>Enough positive examples for algorithm to get a sense of what positive examples are like, future ← positive examples likely to be similar to ones in training set. ←</p> <p>Spam ←</p>

(<http://blog.csdn.net/linuxcumt1>)

Anomaly Detection

anomaly detection skewed class 가 있을 때 사용한다.

Many different **types** of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like

Future anomalies may look nothing like any of the anomalous examples we've seen so far

보면 알겠지만 *anomaly*가 굉장히 다양할 수 있기 때문에 *anomaly*를 특정 형태로 구분짓는 알고리즘을 쓰긴 좀 힘들다.

게다가, 가지고 있는 데이터 셋에서 보지 못했던 새로운 종류의 *anomaly*가 나올 수도 있다.

Supervised Learning

positive, negative example 이 많을 때 사용한다.

Enough positive examples for algorithms to get a sense of what positive examples are like, future positive example likely to be similar to ones in training set

*supervised learning*에서 *positive example*은 어떤 특정 형태기 때문에, 미래에 발견할 *positive example*도 비슷한 형태라 생각될 때 사용한다.

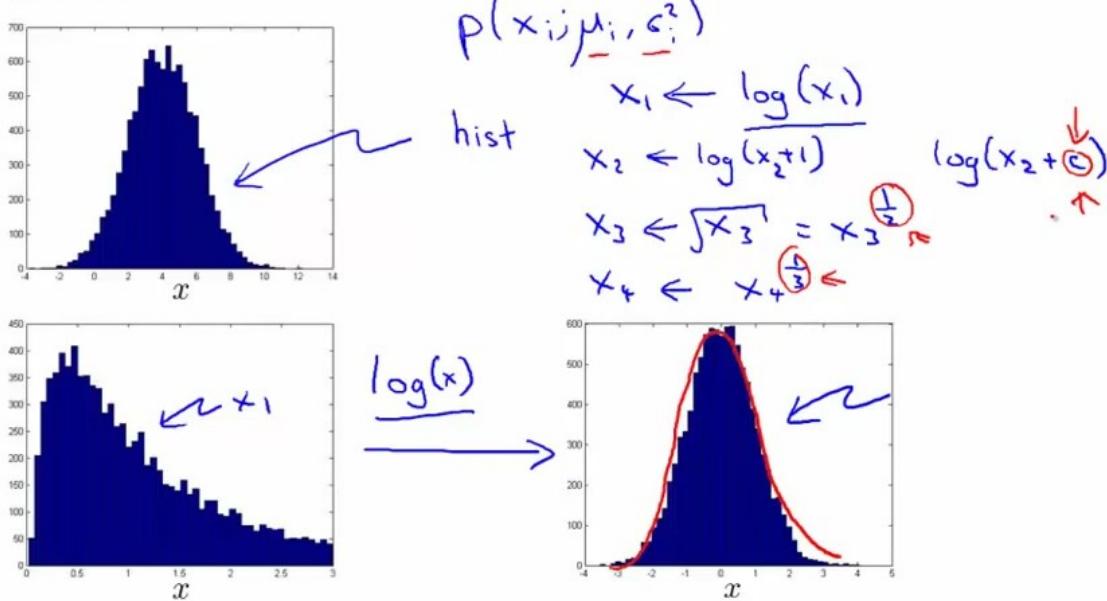
*SPAM filtering*에서는 다양한 타입의 *positive example*이 있어도, 우리가 충분한 양의 *positive example*이 있기 때문에 커버할 수 있어 *supervised learning*을 사용한다.

Anomaly detection	vs.	Supervised learning
→ • Fraud detection	$y=1$	• Email spam classification
→ • Manufacturing (e.g. aircraft engines)		• Weather prediction (sunny/rainy/etc).
→ • Monitoring machines in a data center		• Cancer classification
⋮		⋮

(<http://blog.csdn.net/linuxcumt1>)

Choosing What Features to Use

Non-gaussian features



(<http://blog.csdn.net/linuxcumt1>)

*feature*의 분포가 가우시안이면 고맙지만, 아닐 경우 변환이 필요하다. 왼쪽 아래 분포에 로그를 써우면, 가우시안 분포 비슷하게 보인다.

다른 방법으로는 $\log(x_2 + c)$, $\sqrt{x_3}$ 등등이 있다.

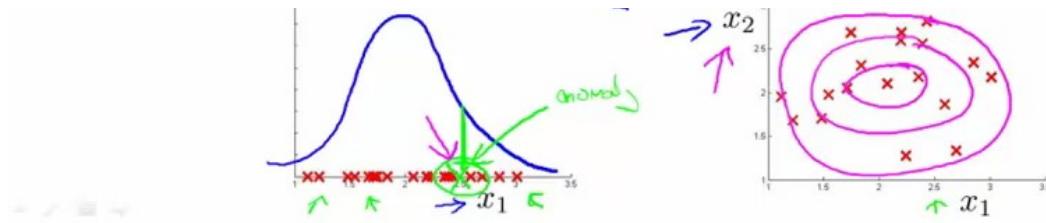
→ Error analysis for anomaly detection

- [Want $p(x)$ large for normal examples x .
p(x) small for anomalous examples x .]

Most common problem:

- [$p(x)$ is comparable (say, both large) for normal and anomalous examples]





흔한 예리는 $p(x)$ 가 *normal, anomalous*에 대해서 모두 높은 경우인데, 슬라이드의 아래쪽에서 볼 수 있듯이 x_2 라는 *feature*를 만들어서 *anomaly*를 발견하는 알고리즘을 만들 수 있다.

→ Monitoring computers in a data center

→ Choose features that might take on unusually large or small values in the event of an anomaly.

→ x_1 = memory use of computer

→ x_2 = number of disk accesses/sec

→ x_3 = CPU load ←

→ x_4 = network traffic ←

$$x_5 = \frac{\text{CPU load}}{\text{network traffic}}$$

$$x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$$

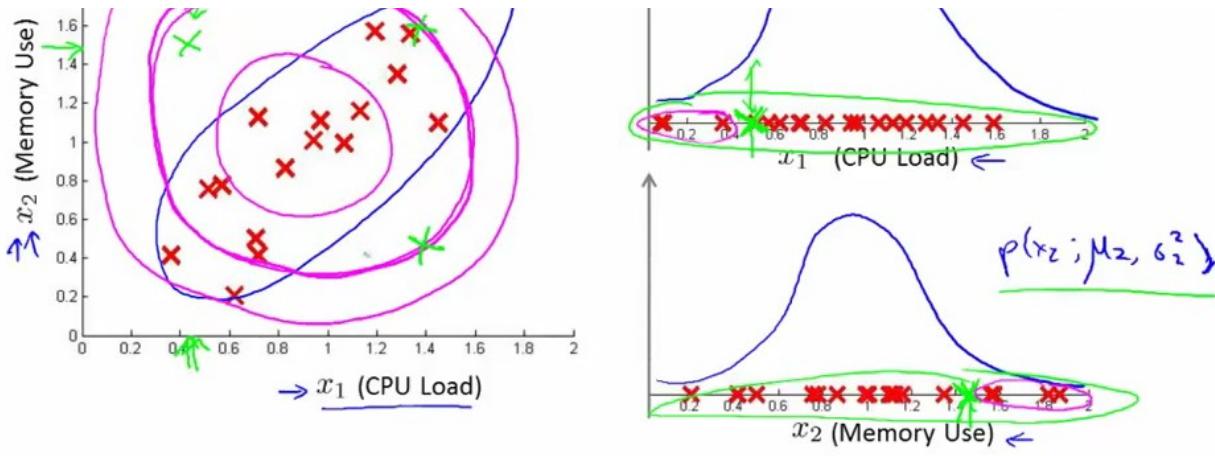
(<http://blog.csdn.net/linuxcumt1>)

*anomaly*를 위한 *feature*를 고를 때 특이하게 높거나, 낮을 수 있는 것을 고르면 된다. 데이터 센터 예제에서는 *CPU load / network traffic* 등이 있을 수 있다. 네트워크 트래픽이 낮은데 *CPU load* 가 높다면 확실히 *anomaly*기 때문이다.

Multivariate Gaussian Distribution

Motivating example: Monitoring machines in a data center





(<http://blog.csdn.net/linuxcumt1>)

feature 를 *CPU load, memory use* 로 했을 때 낮은 CPU 부하에도 메모리 사용량이 높으면 *anomaly* 라 볼 수 있다.

그런데, 슬라이드의 왼쪽 그림에서 녹색으로 표시한 *anomaly*는 지금까지 설명했던 알고리즘으로 찾기가 힘들다. 적당한 수준의 *memory use* 와 그리 낮지 않은 *cpu load*를 가지기 때문이다.

실제 *normal example*이 타원형이기 때문에, 원으로 *anomaly*를 찾기는 어렵다.

Multivariate Gaussian (Normal) distribution

$\rightarrow x \in \mathbb{R}^n$. Don't model $p(x_1), p(x_2), \dots$, etc. separately.

Model $p(x)$ all in one go.

Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (covariance matrix)

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

$|\Sigma| = \text{determinant of } \Sigma \quad | \det(\text{Sigma})|$

따라서 $p(x_1)p(x_2)\dots$ 을 이용한 모델 말고 다른 방법으로 모델을 만들어야 한다.

u 를 n 벡터라 하고, Σ 를 u 의 covariance matrix 라 하자. 그러면

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right)$$

여기서 $|\Sigma|$ 는 Σ 의 행렬식인데, 여기를 참고하자.

- 행렬
- 행렬식
- 행렬식의 기하학적 의미
- 행렬식과 기하학적 활용

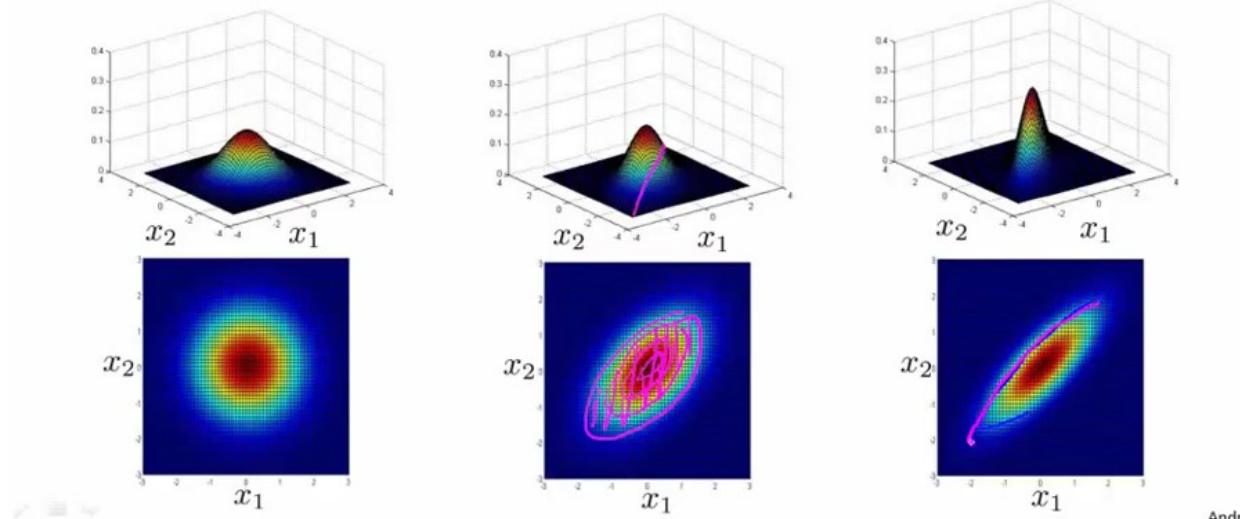
이제 위 식을 이용해서 나온 $p(x)$ 를 3차원, 2차원으로 보면

Multivariate Gaussian (Normal) examples

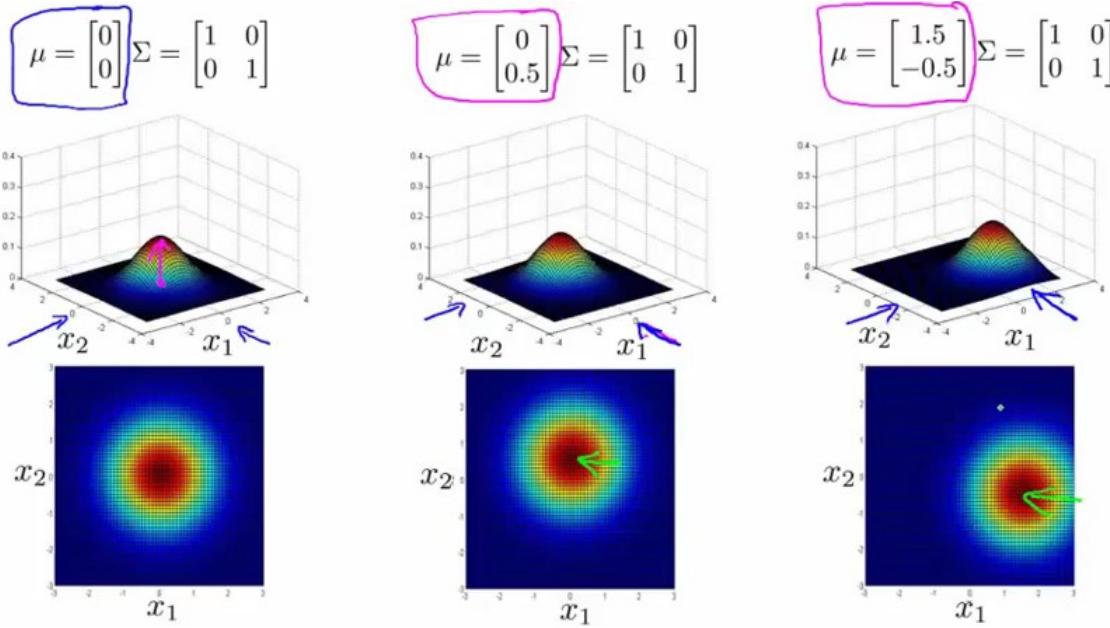
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

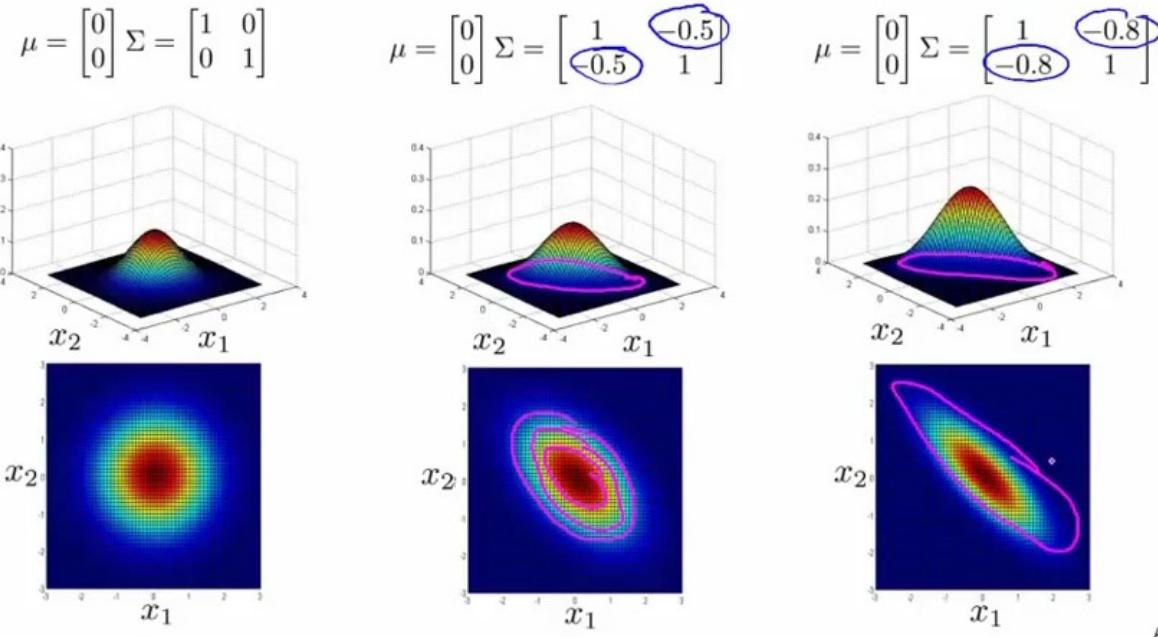
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$



Multivariate Gaussian (Normal) examples



Multivariate Gaussian (Normal) examples

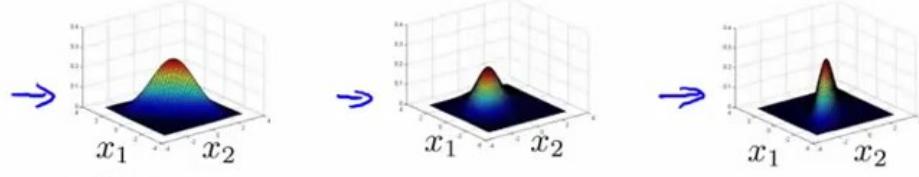


Multivariate Gaussian (Normal) distribution

Parameters $\underline{\mu, \Sigma}$

$$\mu \in \mathbb{R}^n \quad \Sigma \in \mathbb{R}^{n \times n}$$

$$\rightarrow p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$



Parameter fitting:

Given training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\} \leftarrow x \in \mathbb{R}^n$

$$\boxed{\mu} = \frac{1}{m} \sum_{i=1}^m x^{(i)} \quad \boxed{\Sigma} = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

(<http://blog.csdn.net/linuxcumt1>)

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

Anomaly detection with the multivariate Gaussian

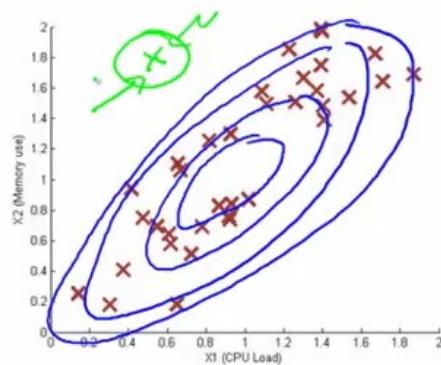
1. Fit model $p(x)$ by setting

$$\begin{cases} \mu = \frac{1}{m} \sum_{i=1}^m x^{(i)} \\ \Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T \end{cases}$$

2. Given a new example x , compute

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

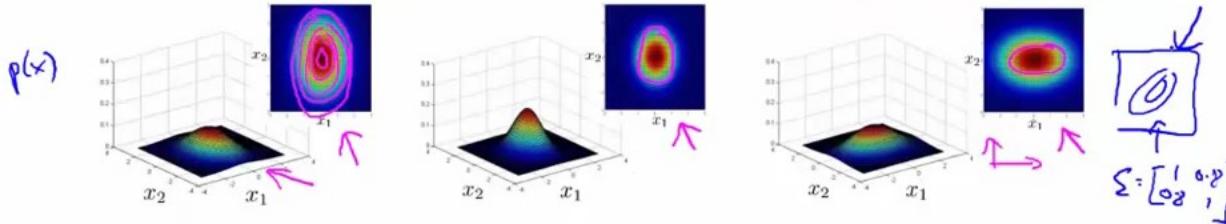
Flag an anomaly if $p(x) < \varepsilon$



u , Σ 를 찾아 $p(x)$ 를 만들고, 테스트 데이터에 대해 $p(x) < e$ 인지 비교 한다.

Relationship to original model

Original model: $p(x) = p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$



Corresponds to multivariate Gaussian

$$\rightarrow p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

where $\Sigma = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$

Andrew Ng

original model 은 multivariate model 에서 각 feature 간 상관 관계가 없는 (독립), 즉 covariance matrix 가 diagonal matrix 인 경우다.

→ Original model

$$p(x_1; \mu_1, \sigma_1^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$$

Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values.
 $\rightarrow X_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{memory}}$

→ Computationally cheaper
 (alternatively, scales better to large n) $n=10,000, \quad n=100,000$

OK even if m (training set size) is small

vs. → Multivariate Gaussian

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

→ Automatically captures correlations between features

$$\Sigma \in \mathbb{R}^{n \times n} \quad \Sigma^{-1}$$

Computationally more expensive

$$\rightarrow \Sigma \sim \frac{n^2}{2}$$

Must have $m > n$ or else Σ is non-invertible

(<http://blog.csdn.net/linuxcumt1>)

- Original model

수동으로 *feature* 를 만들 때 사용할 수 있다. 또는 적은 연산을 원할 때, 다시 말해서 n 이 커서 연산이 무지막지하게 클 때 좋다.

m 이 작아도 쓸 수 있다.

- Multivariate Gaussian

계산 비용이 비싸지만, 자동으로 *feature* 간 상관관계를 모델에 포함시킨다.

Sigma 가 invertible 이기 위해서는 $m > n$ 이어야 한다. 실제로는 m 이 n 보다 훨씬 클 때 사용하는 경우가 많다. (e.g. $m \geq 10n$)

만약에 $m > n$ 인데, *Sigma* 가 non-invertible 이면 redundant feature 가 있는 경우니 확인해 보자. (흔한 오류라고 함)

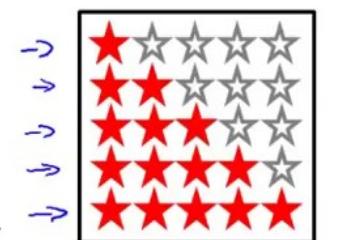
Recommender System

Example: Predicting movie ratings

→ User rates movies using one to five stars
zero

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	6
Romance forever	5	?	0	0
Cute puppies of love	?	5	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

$n_u = 4$ $n_m = 5$



$\rightarrow n_u = \text{no. users}$
 $\rightarrow n_m = \text{no. movies}$
 $\rightarrow r(i, j) = 1$ if user j has rated movie i
 $y^{(i,j)}$ = rating given by user j to movie i
 (defined only if $r(i, j) = 1$)

Content Based Recommendations

Content-based recommender systems

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	$n_u = 4, n_m = 5$	$x^{(i)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$
$x^{(1)}$ Love at last 1	5	5	0	0	x_1 (romance) $\rightarrow 0.9 \rightarrow 0$	
$x^{(2)}$ Romance forever 2	5	?	?	0	x_2 (action) $\rightarrow 1.0 \rightarrow 0.01$	
$x^{(3)}$ Cute puppies of love 3	?	4.95	4	?	x_3 (romance) $\rightarrow 0.99 \rightarrow 0$	
$x^{(4)}$ Nonstop car chases 4	0	0	5	4	x_4 (action) $\rightarrow 0.1 \rightarrow 1.0$	
$x^{(5)}$ Swords vs. karate 5	0	0	5	?	x_5 (romance) $\rightarrow 0 \rightarrow 0.9$	$n=2$

→ For each user j , learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

$$\theta^{(1)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix} \leftrightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad (\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$$

(<http://blog.csdn.net/linuxcumt1>)

위 슬라이드는 유저 j 로 부터 $\theta^{(j)}$ 를 얻어, feature x 와 곱함으로써 linear regression 문제로 변경했다.

Problem formulation

- $r(i, j) = 1$ if user j has rated movie i (0 otherwise)
- $y^{(i,j)}$ = rating by user j on movie i (if defined)

- $\theta^{(j)}$ = parameter vector for user j
- $x^{(i)}$ = feature vector for movie i
- For user j , movie i , predicted rating: $\underline{(\theta^{(j)})^T(x^{(i)})}$ $\theta^{(j)} \in \mathbb{R}^{n+1}$
- $m^{(j)}$ = no. of movies rated by user j
- To learn $\underline{\theta^{(j)}}$:

$$\min_{\theta^{(j)}} \frac{1}{2m^{(j)}} \sum_{i: r(i,j)=1} ((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)})^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$\theta^{(j)}$ 는 어떻게 훈련시킬까?

$$\min_{\theta^{(j)}} \sum_{i: r(i,j)=1} \frac{1}{2m^{(j)}} [(\theta^{(j)})^T x^{(i)} - y^{(i,j)}]^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^n (\theta_k^{(j)})^2$$

여기서 $r^{(j)}$ 는 유저 j 에 의해 점수를 받은 영화의 수인데, 어차피 상수이므로 제거하면

$$\min_{\theta^{(j)}} \sum_{i: r(i,j)=1} \frac{1}{2} [(\theta^{(j)})^T x^{(i)} - y^{(i,j)}]^2 + \frac{\lambda}{2} \sum_{k=1}^n (\theta_k^{(j)})^2$$

Optimization objective:

To learn $\underline{\theta^{(j)}}$ (parameter for user j):

$$\rightarrow \min_{\theta^{(j)}} \underbrace{\frac{1}{2} \sum_{i: r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2}_{\text{loss term}} + \underbrace{\frac{\lambda}{2} \sum_{k=1}^n (\theta_k^{(j)})^2}_{\text{regularization term}}$$

To learn $\underline{\theta^{(1)}}, \underline{\theta^{(2)}}, \dots, \underline{\theta^{(n_u)}}$:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \left[\sum_{j=1}^{n_u} \sum_{i: r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2 \right]$$

$\Theta^{(1)}, \dots, \Theta^{(n_u)}$

(<http://blog.csdn.net/linuxcumt1>)

이 때 각 유저마다의 $\theta^{(j)}$ 를 합 해 최소화 시키는 방식으로 전체 θ 를 훈련시킬 수 있다.

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}}$$

$$= \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i: r(i,j)=1} [(\theta^{(j)})^T x^{(i)} - y^{(i,j)}]^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

Optimization algorithm:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

↙

$J(\theta^{(1)}, \dots, \theta^{(n_u)})$

Gradient descent update:

$$\begin{aligned} \theta_k^{(j)} &:= \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} \quad (\text{for } k = 0) \\ \theta_k^{(j)} &:= \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} \underbrace{((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)}}_{\frac{\partial J}{\partial \theta_k^{(j)}}} + \lambda \theta_k^{(j)} \right) \quad (\text{for } k \neq 0) \end{aligned}$$

~~$\frac{1}{m+1}$~~ $\frac{\partial J}{\partial \theta_k^{(j)}}$ $J(\theta^{(1)}, \dots, \theta^{(n_u)})$



(<http://blog.csdn.net/linuxcumt1>)

gradient descent \Leftarrow

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i: r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)}) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i: r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \quad (\text{for } k \neq 0)$$

Collaborative Filtering

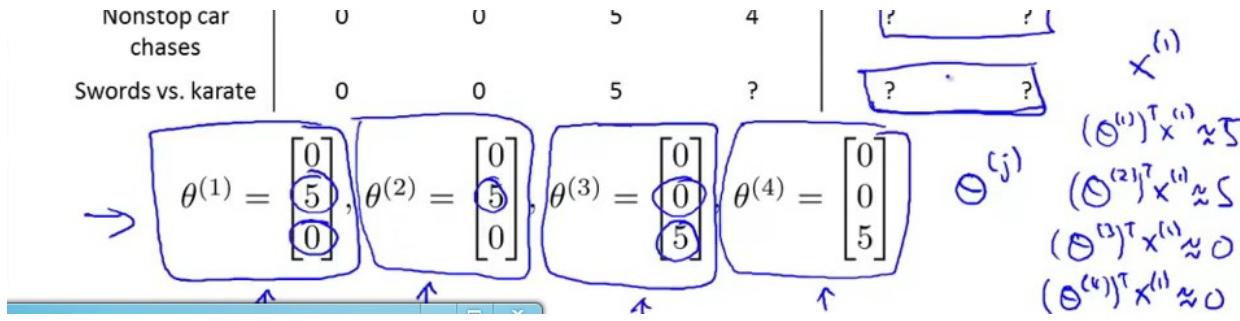
Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)	$x_0 = 1$
Love at last	5	5	0	0	1.0	0.0	
Romance forever	5	?	?	0	?	?	
Cute puppies of love	?	4	0	?	?	?	
...	-	-	-	-	-	-	

$X^{(1)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 5 & 5 & 0 & 0 \\ ? & ? & ? & 0 \\ ? & 4 & 0 & ? \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$

$X^{(2)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1.0 & 0.0 & 1.0 & 0.0 \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$

$X^{(3)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1.0 & 0.0 & 1.0 & 0.0 \\ 1.0 & 0.0 & 1.0 & 0.0 \\ ? & ? & ? & ? \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$



(<http://blog.csdn.net/linuxcumt1>)

content-based recommendation에서 feature를 구하긴 사실 어려운 일이다. 누가 이 영화가 얼마만큼 로맨스고, 아닌지를 판별해줄까?

문제를 좀 변경해서, 만약에 유저로부터 $\theta^{(j)}$ 를 얻어낼 수 있다면 그 정보로부터 feature $x^{(i)}$ 를 추출할 수 있다. 왜냐하면

$(\theta^{(j)})^T * x^{(i)} \sim y^{(i, j)}$ 이기 때문이다.

Optimization algorithm

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, to learn $x^{(i)}$:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^n (x_k^{(i)})^2$$

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, to learn $x^{(1)}, \dots, x^{(n_m)}$:

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

$x^{(i)}$ 를 얻기 위해,

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} [(\theta^{(j)})^T x^{(i)} - y^{(i,j)}]^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

$$2 \sum_{i=1}^m \sum_{j: r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \lambda \sum_{k=1}^n (\theta_k^{(j)})^2$$

Collaborative filtering

[Given $x^{(1)}, \dots, x^{(n_m)}$ (and movie ratings),
can estimate $\theta^{(1)}, \dots, \theta^{(n_u)}$]

$$\begin{matrix} r^{(i,j)} \\ y^{(i,j)} \end{matrix}$$

[Given $\theta^{(1)}, \dots, \theta^{(n_u)}$,
can estimate $x^{(1)}, \dots, x^{(n_m)}$]

Guess $\Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \dots$

(<http://blog.csdn.net/linuxcumt1>)

- theta 가 주어지면 x 를 훈련할 수 있고
- x 가 주어지면 theta 를 훈련할 수 있다.

따라서 최초의 랜덤 theta 에 대해 x 를 훈련하고, 다시 theta 를 훈련하고, 반복하면 된다.

Collaborative filtering optimization objective $r^{(i,j)} : r^{(i,j)} = 1$

→ Given $x^{(1)}, \dots, x^{(n_m)}$, estimate $\theta^{(1)}, \dots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i: r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

→ Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, estimate $x^{(1)}, \dots, x^{(n_m)}$:

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j: r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

Minimizing $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$ simultaneously:

$$\begin{aligned} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) &= \frac{1}{2} \sum_{(i,j): r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2 \\ &\min_{\substack{x^{(1)}, \dots, x^{(n_m)} \\ \theta^{(1)}, \dots, \theta^{(n_u)}}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) \end{aligned}$$

Andrew Ng

theta 와 **x** 를 반복해서 훈련시키는 것보다, 동시에 훈련시키는 것이 좀 더 효율적이다. 따라서

$$J(x^{(j)}, \dots, x^{(n_m)}, \theta^{(i)}, \dots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j): r(i,j)=1} [(\theta^{(j)})^T x^{(i)} - y^{(i,j)}]^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

를 최소화 시키면 된다. 참고로 **x_0** 은 collaborative filtering에서 필요가 없다. 알고리즘 자체가 feature 를 직접 찾아내니 hard coded 된 feature 는 사용하지 않는다.

Collaborative filtering algorithm

→ 1. Initialize $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$ to small random values.

→ 2. Minimize $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \dots, n_u, i = 1, \dots, n_m$:

$$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right) \quad \frac{\partial J}{\partial x_k^{(i)}}$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right) \quad \frac{\partial J}{\partial \theta_k^{(j)}}$$

3. For a user with parameters θ and a movie with (learned) features x , predict a star rating of $\theta^T x$.

$$(\theta^{(i)})^T (x^{(i)})$$

(<http://blog.csdn.net/linuxcumt1>)

(1) **x**, **theta** 를 작은 값으로 초기화 한다.

이는 symmetry breaking 을 하기 위함이다. 작은 랜덤값들로 초기화 하여 $x^{(i)}$ 가 서로 다른 값들을 가지도록 도와준다.

(2) cost function **J** 를 gradient descent 등으로 최소화 시킨다.

$$x_k^{(i)} := x_k^{(i)} - \alpha \sum_{j:r(i,j)=1} [(\theta^{(j)})^T x^{(i)} - y^{(i,j)}] \theta_k^{(j)} + \lambda x_k^{(i)}$$

$$j: \overbrace{r(i,j)=1}$$

$$\theta_k^{(i)} := \theta_k^{(i)} - \alpha \sum_{j: r(i,j)=1} [(\theta^{(j)})^T(x^{(i)}) - y^{(i,j)}]_k^{(i)} + \lambda \theta_k^{(j)}$$

(3) 유저의 parameter `theta` 와 영화의 `feature x`에 대해 `theta^T * x` 를 이용해 예측하면 된다.

Vectorization: Low Rank Matrix Factorization

Collaborative filtering

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	
Love at last	5	5	0	0	
Romance forever	5	?	?	0	
Cute puppies of love	?	4	0	?	
Nonstop car chases	0	0	5	4	
Swords vs. karate	0	0	5	?	

$n_m = 5$
 $n_u = 4$

↑ ↑ ↑ ↑ ↑ $y_{(i,j)}$

Collaborative filtering

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

$$\begin{aligned} X \odot \Theta^T &\leftarrow (\theta^{(1)})^T(x^{(1)}) \\ \text{Predicted ratings: } (i,j) &\rightarrow \\ Y &= \begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \dots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \dots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \dots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix} \end{aligned}$$

$$\begin{aligned} & \left[\begin{array}{c} -(x^{(1)})^T \\ -(x^{(2)})^T \end{array} \right] \\ \Theta &= \left[\begin{array}{c} -(\theta^{(1)})^T \\ -(\theta^{(2)})^T \end{array} \right] \end{aligned}$$

$$\xrightarrow{x} \left[- (x^{(n,m)})^T \right] \quad \xrightarrow{\theta} \left[- (\Theta^{(n,u)})^T \right]$$

Low rank matrix factorization

Andrev

collaborative filtering 은 low rank matrix factoriazation 이라 부르기도 한다. 위 슬라이드처럼 x, Θ 를 구성하고 $x * \Theta^T$ 를 구하면 된다.

Finding related movies

For each product i , we learn a feature vector $\underline{x^{(i)}} \in \mathbb{R}^n$.

$\rightarrow x_1 = \text{romance}, x_2 = \text{action}, x_3 = \text{comedy}, x_4 = \dots$

How to find $\underline{\text{movies } j}$ related to $\underline{\text{movie } i}$?

small $\|x^{(i)} - x^{(j)}\| \rightarrow \text{movie } j \text{ and } i \text{ are "similar"}$

5 most similar movies to movie i :

Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.

(<http://blog.csdn.net/linuxcumt1>)

low rank matrix factorization 을 이용해서 feature 를 찾으면, 두 영화 i, j 가 얼마나 유사한지 $\|x^{(i)} - x^{(j)}\|$ 으로 판단할 수 있다.

Implementation Detail: Mean Normalization

Users who have not rated any movies

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)	
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?

\downarrow
 \downarrow
 \downarrow
 \downarrow

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \end{bmatrix}$$

Nonstop car chases	0	0	5	4	<u>?</u>	0	~0	~5	~0	<u>?</u>
Swords vs. karate	0	0	<u>5</u>	?	<u>?</u>	0	0	5	0	<u>?</u>

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{(i,j): r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

$\Theta^{(s)} \in \mathbb{R}^2$ $\Theta^{(s)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$(\Theta^{(s)})^T x^{(i)} = 0$

만약 위 슬라이드의 Eve 처럼 아무 영화도 평가 안한 사람에게는, theta 가 0 으로 나온다. (첫번째 term 이 0 이고, regularization term 은 theta 를 최소화한다.)

그렇게 되면, 어떤 영화도 높은 rating 을 받을 수 없으므로 ($\theta^{(s)T} * x^{(i)}$). 추천할 거리가 없다. 이건 별로 좋은 상황이 아닌데, mean normalization 을 이용하면 이 문제를 해결할 수 있다.

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? & 2.5 \\ 5 & ? & ? & 0 & ? & 2.5 \\ ? & 4 & 0 & ? & ? & 2 \\ 0 & 0 & 5 & 4 & ? & \vdots \\ 0 & 0 & 5 & 0 & ? & \vdots \end{bmatrix}$$

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user j , on movie i predict:

$$\rightarrow (\Theta^{(s)T} (x^{(i)}) + \mu_i)$$

learn $\Theta^{(s)}, x^{(i)}$

User 5 (Eve):

$$\Theta^{(s)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\underbrace{(\Theta^{(s)T} (x^{(i)})}_{=0} + \boxed{\mu_i}$$

(<http://blog.csdn.net/linuxcumt1>)

mean normalized 데이터를 이용하면, 추천 안한 사람이 $\theta = 0$ 을 갖더라도, 남들이 추천한 선호도 u 에 따라서 영화를 추천받을 수 있다.

$\ell_A(j) \setminus T \ell_m(i) \cup \dots$

$$(U^{(i)})_{j} = \frac{(U^{(i)})_j - \mu_i}{\sigma_i}$$

잘보면 *feature scaling* 과는 다르게 특정 *range*로 나누질 않는데, 이건 이미 *rating* 자체가 일정 범위 [1-5]를 갖기 때문이다.

References

- (1) *Machine Learning* by Andrew NG
- (2) <http://blog.csdn.net/linuxcumt>
- (3) <http://blog.csdn.net/abcjennifer>
- (4) <http://ghebook.blogspot.com>
- (5) <http://darkpgmr.tistory.com>

0 Comments lambda 1 Login ▾

Heart Recommend Share Sort by Best ▾

 Start the discussion...

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

✉ Subscribe D Add Disqus to your site [Add Disqus](#) 🔒 Privacy

comments powered by Disqus

